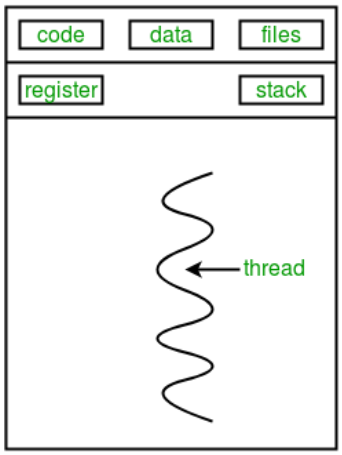
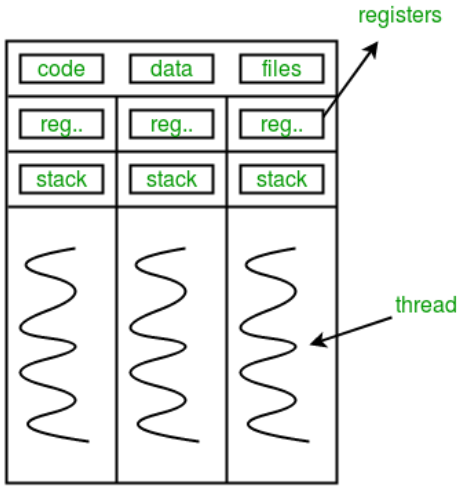
# Ejercicios de la UD01




single-threaded process




multithreaded process

# 1. Ejercicios

1. Cree un proyecto llamado `ProcessListPNG` con un programa que le pida al usuario que introduzca una ruta (por ejemplo, `/micarpeta/fotos`), y luego inicie un proceso que imprima una lista de todas las imágenes PNG encontradas en esta ruta. Intenta hacerlo de forma recursiva (ya sea con un comando del sistema operativo o con tu propio script).
2. Cree un programa Java llamado ReadName.java que reciba un nombre de los argumentos main() y lo muestre en la pantalla. Utiliza System.exit(1) para una terminación correcta del programa y System.exit(-1) en caso de que no se hayan ingresado los argumentos correctos en main().

   A continuación, cree un programa similar al siguiente:

```java
import java.io.*;

/**
 *
 * @author David Martínez (wwww.martinezpenya.es|iesmre.com)
 */
public class Test {

    public static void main(String[] args) throws IOException {
        //create File Object where the example is located
        File folder = new File(".\\bin");
        //process to execute
        ProcessBuilder pb = new ProcessBuilder("java", "example");
        //move to that folder
        pb.directory(folder);
        System.out.format("Working folder: %s%n", pb.directory());
        //run the process
        Process p = pb.start();
        //get the output of the process
        try {
            InputStream is = p.getInputStream();
            int c;
            while ((c = is.read()) != -1) {
                System.out.println((char) c);
            }
            is.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

para ejecutar `ReadName.java`. Utiliza el método waitFor() para verificar el valor de salida del proceso en ejecución. Prueba la ejecución del programa dando valor a los argumentos de main() y sin darle valor. ¿Qué devuelve waitFor() en un caso y en otro?

3. Cree un proyecto llamado `ProcessKillNotepad` con un programa que inicie el bloc de notas o cualquier editor de texto similar desde su sistema operativo. Luego, el programa esperará 10 segundos a que finalice el subproceso y, transcurrido ese tiempo, será destruido. Para dormir 10 segundos, usa esta instrucción:

```
Thread.sleep(milliseconds);
```

4. Cree un proyecto llamado `Lavavajillas`. Vamos a simular un proceso de lavado de platos en casa, cuando alguien lava los platos y alguien más los seca. Crea las siguientes clases:
    - Una clase `Plato` con solo un atributo entero: el número del plato (para identificar los diferentes platos).
    - Una clase `DishPile` que almacenará hasta 5 platos. Tendrá un método de lavado que pondrá un plato en la pila (si hay espacio disponible), y un método de secado que sacará un plato de la pila (si hay alguno). Tal vez necesite un parámetro Plato en el método de lavado para agregar un plato a la pila.
    - Un hilo `Washer` que recibirá un número N como parámetro, y un objeto `DishPile`. En su método run, pondrá (lavará) N platos en la pila, con una pausa de 50ms entre cada plato.
    - Un hilo `Dryer` que recibirá un número N como parámetro, y un objeto `DishPile`. En su método de ejecución, sacará (secará) N platos de la pila, con una pausa de 100 ms entre cada plato.
    - La clase `main` creará el objeto `DishPile` y un hilo de cada tipo (`Washer` y `Dryer`). Tendrán que lavar/secar 20 platos coordinadamente, por lo que el resultado debe ser algo así:

```
Washed dish #1, total in pile: 1
Drying dish #1, total in pile: 0
Washed dish #2, total in pile: 1
Drying dish #2, total in pile: 0
Washed dish #3, total in pile: 1
Washed dish #4, total in pile: 2
Drying dish #4, total in pile: 1
Washed dish #5, total in pile: 2
Washed dish #6, total in pile: 3
Drying dish #6, total in pile: 2
Washed dish #7, total in pile: 3
...
```

# 2. Actividades

1. El objetivo de esta actividad es comprender el problema que plantea la programación concurrente cuando no se aplican métodos de protección en tramos críticos.
   Disponemos de un sistema que controla las monedas que entran en los parquímetros cada hora para controlar el momento del día de mayor tráfico en el parking.
   Para ello los parquímetros tienen dos procesos ejecutándose en su sistema:

   - Un proceso contador, que contará las monedas que entren en el parquímetro.
   - Otro proceso que cada hora envía el número de monedas a la central y pone a cero el número de monedas.

   Hay una variable compartida que tendrá el número de monedas en el parquímetro. Son dos procesos que se ejecutan concurrentemente, por lo que es posible que en un momento dado se ejecuten al mismo tiempo.

   Indica los posibles órdenes de ejecución de las instrucciones 1, 2 y 3 de los dos procesos si se ejecutan concurrentemente. Indique también cuál de los posibles órdenes sería el correcto.

   1. `sendHeadOffice(moneda)`
   2. `moneda=0`
   3. `moneda=moneda+1`

   El código de los procesos sería el siguiente:

```
void enviaNumeroModenes(){
  while (true){
    enviarOficinaCentral(moneda)
    moneda=0
    }
       }
   void comptadorModenes(){
    while (entradaModena==true)
        moneda=moneda+1
     }
   }
 }
```

2. Di si las siguientes afirmaciones son verdaderas (V) o falsas (F):

| No | Pregunta | V | F |
|---|---|---|---|
| 1 | Un proceso siempre tiene un estado | | |
| 2 | Un proceso y una aplicación son sinónimos | | |
| 3 | En la programación concurrente un procesador puede ejecutar diferentes procesos al mismo tiempo | | |

3. Responda las siguientes preguntas:

   - Escribir alguna característica de un programa concurrente.
   - ¿Cuál es la ventaja de la concurrencia en sistemas monoprocesador?
   - ¿Cuáles son las diferencias entre multiprogramación y multihilo?

- ¿Cuáles son los dos principales problemas inherentes a la programación concurrente?

4. Vaya a la siguiente URL https://hpc.llnl.gov/documentation/tutorials/introduction-parallel-computing-tutorial y responda las siguientes preguntas:
   - Mencionar algunas características de la computación en serie.
   - Mencionar algunas características de la computación paralela.
   - Áreas en las que se utiliza la computación paralela.
   - ¿Cómo utiliza el proyecto SETI@home la computación paralela?
5. ¿Cuál de los siguientes sistemas operativos no es multitarea?

   a) Unix.

   b)Linux.

   c) Windows 10.

   d) MS-DOS.
6. En informática se entiende por multitarea:

   a) La capacidad de una computadora para realizar tareas complejas.

   b) La capacidad de una computadora para ejecutar un programa tras otro.

   c) La capacidad de una computadora para realizar tareas sincronizadas entre sí.

   d) La capacidad de una computadora para ejecutar varios programas al mismo tiempo.
7. ¿Qué elemento del ordenador se encarga de gestionar la multitarea?

   a) El lenguaje de programación elegido para crear los programas.

   b) El sistema operativo.

   c) El sistema de archivos.

   d) Ninguna de las respuestas anteriores es correcta.
8. ¿Qué afirmación sobre lenguajes de programación compilados no es correcta?

   a) En condiciones normales son más rápidos que los interpretados.

   b) Requieren una compilación específica para cada sistema operativo.

   c) Si el lenguaje es estándar, la misma compilación funciona para todos los sistemas operativos.

   d) Se debe compilar el código fuente para obtener el código binario a ejecutar.
9. El identificador del proceso suele identificarse por las siglas:

   a) CPU.

   b) DPI.

   c) DPI.

   d) TIY.
10. Procesamiento ejecutándose en diferentes computadoras independientes pero conectadas
    y sincronizado se llama:

    a) Distribuido.

    b) Concurrentes.

    c) Paralelo.

d) Monohilo.

11. ¿Cuál de los siguientes no es un objetivo del planificador de procesos del sistema operativo?

    a) Maximizar el rendimiento del sistema.

    b) Maximizar los tiempos de respuesta.

    c) Maximizar la equidad en la distribución de los recursos.

    d) Minimizar los tiempos de espera.

12. ¿Cuál es el estado al que puede ir un proceso que está en estado Listo?

    a) Nuevo.

    b) Bloqueado.

    c) Correr.

    d) Terminado.

13. ¿Qué afirmación sobre la clase Java Runtime es incorrecta?

    a) Permite lanzar un proceso indicando los parámetros de ejecución.

    b) Permite esperar la finalización del proceso iniciado.

    c) Permite conocer el valor de las variables internas del proceso lanzado.

    d) Permite obtener el estado de finalización del proceso iniciado.

14. ¿Cómo llamas al método de la clase Java Process que te hace esperar a que se inicie el proceso?

    a) `sleep()`.
    b) `finish()`.
    c) `waitFor()`.
    d) `waitEnd()`.

# 3. Actividades de aplicación

1. Explicar las diferencias entre los lenguajes de programación interpretados y compilados.
2. Describir las diferencias entre programa y proceso.
3. Explicar en qué consiste la programación distribuida.
4. Encuentra las diferencias entre ejecutar dos procesos o realizar una bifurcación.
5. Diseñar un sistema basado en una base de datos para intercambiar información entre dos procesos.
6. Diseñar un sistema basado en archivos de texto para sincronizar procesos.
7. Decida qué sistema de sincronización podría usarse para sincronizar los procesos que se ejecutan en computadoras independientes conectadas a través de Internet.
8. Explique por qué los hilos consumen menos recursos que los procesos.
9. Explique qué tarea realiza el programador de procesos dentro del sistema operativo.
10. Diseñe un planificador de procesos simple. Determina cómo se asigna el tiempo de CPU a los diversos procesos que administra para que todos puedan proceder más o menos simultáneamente.
11. Elaborar la lista de valores y sus descripciones que puede devolver la ejecución de un proceso que tiene que acceder a una base de datos, una computadora a través de la red e Internet.
12. Escriba un programa Java que ejecute el navegador web **Firefox**. Utiliza la clase `Runtime`. Intenta que el navegador abra directamente una página web.

# 4. Fuentes de información

- Wikipedia
- Programación de servicios y procesos - FERNANDO PANIAGUA MARTÍN [Paraninfo]
- Programación de Servicios y Procesos - ALBERTO SÁNCHEZ CAMPOS [Ra-ma]
- Programación de Servicios y Procesos - Mª JESÚS RAMOS MARTÍN - [Garceta] (1ª y 2ª Edición)
- Programación de servicios y procesos - CARLOS ALBERTO CORTIJO BON [Síntesis]
- Programació de serveis i processos - JOAR ARNEDO MORENO, JOSEP CAÑELLAS BORNAS i JOSÉ ANTONIO LEO MEGÍAS [IOC]
- GitHub repositories:
    - https://github.com/ajcpro/psp
    - https://oscarmaestre.github.io/servicios/index.html
    - https://github.com/juanro49/DAM/tree/master/DAM2/PSP
    - https://github.com/pablohs1986/dam_psp2021
    - https://github.com/Perju/DAM
    - https://github.com/eldiegoch/DAM
    - https://github.com/eldiegoch/2dam-psp-public
    - https://github.com/franlu/DAM-PSP
    - https://github.com/ProgProcesosYServicios
    - https://github.com/joseluisgs
    - https://github.com/oscarnovillo/dam2_2122
    - https://github.com/PacoPortillo/DAM_PSP_Tarea02_La-Cena-de-los-Filosofos